IN THE SPECIFICATION

On page 13, line 13, replace the entire paragraph with the following paragraph as amended.

The integration between application programs 32 and 34 in the example described above is achieved through the use of a control file such as that of which a portion is illustrated in Figs. 7A and 7B. The illustrated portion of the control file has six columns, the first ~~three~~ four being shown in ~~Fig. 6A~~ Fig. 7A and the final two ~~second three~~ being continued in ~~Fig. 6B~~ Fig. 7B. The meanings of the elements in the columns are described in detail below, but note that some of the element names include references to "LDAP," i.e., the directory function, and others include references to the "e-mail" function. In the example described above in which device 26 includes application programs 32, 34, and 36, the control file would include references not only to the e-mail and directory functions but also the contact manager function represented by the application program 36. The control file can be created using any suitable authoring means, such as a text editor or a spreadsheet program, and the fields can be delimited in any suitable manner such as columns or separating elements with commas or other characters. The control file is loaded into device 26 in essentially the same manner as application programs 32, 34, and 36. As described below, the control file controls how the screens are arranged, what buttons or other user interface controls are displayed, how they are labeled, and the JAVA method associated with each user interface control.

On page 18, line 7, replace the entire paragraph with the following paragraph as amended.

Although the sequence of operation is described in further detail below, when device 26 is initialized by the user by turning it on, logging in, resetting it, or by a similar system startup action, objects are instantiated in accordance with the control file and classes defined by framework 24. Some of these framework classes are shown in the class diagram of Fig. 11. Persons skilled in the art will note that the class names begin with the letter "I" to denote JAVA interface classes rather than implementation classes.

~~IScreenInfo~~ IApplicationInfo class 110 represents a single screen. Components of class 110 can be read from a configuration file (not shown in Fig. 10) at startup. Class 110 refers to an ICommandInfo class 112, an IBusListenerInfo class 114 and one of three types of an IProvider class 116, a view provider, and I/O provider or a storage provider. Providers are explained below.

On page 18, line 17, replace the entire paragraph with the following paragraph as amended.

The screen represented by ~~IScreenInfo~~ IApplicationInfo class 110 corresponds to a group of lines of the control file. In other words, an instance of this class is created in response to the group of lines. Each line of the control file has several columns. Referring to Fig. 7A, note that, for example, the first line of the second group of lines from the top (groups being offset from one another by blank lines) includes "App" in the first column, RootApp.Menu" in the second column, "MainMenu" in the third column and, continuing on Fig. 7B, "com.bonitasoftware.togo.MenuApplication" in the fifth column.